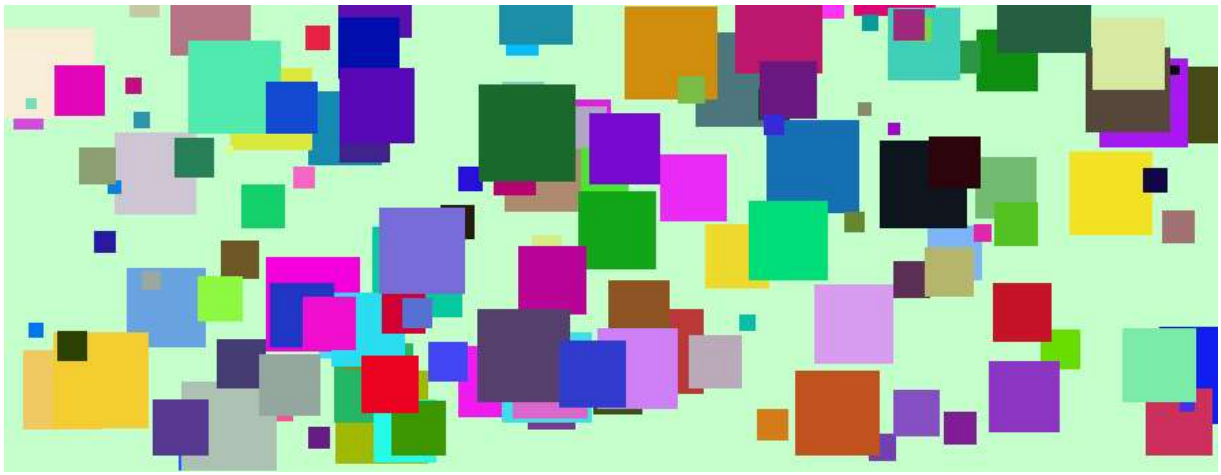
TP : Artleatoire

Dernière mise à jour le : 06/12/2023

Dans ce TP, vous allez utiliser les modules `turtle` et `random` pour générer des oeuvres d'art aléatoires.



Prise en main du module `turtle`

Le module `turtle` est déjà installé dans les versions standards de Python.

> Le code présenté dans la première question vous montre les fonctionnalités qui seront nécessaires dans ce TP. Si vous voulez en savoir davantage, la documentation officielle du module est accessible à cette adresse : https://docs.python.org/fr/3.10/library/turtle.html.

💻 **Question 1** : Créez un répertoire et nommez-le comme bon vous semble, ce sera notre répertoire pour tout ce PT. Créez dans ce dossier un fichier `essais.py` et copiez-y le code ci-dessous :

```
from turtle import *

setup(800, 600)  # taille de fenêtre égale à 800*600 (en pixels)
speed(1)  # de 1 (lent) à 10 (rapide) et 0 le plus rapide

pensize(3)  # largeur du trait : 1 au minimum
forward(100)  # avance de 100
backward(200)  # recule de 200
goto(0, 100)  # se déplace aux coordonnées 0, 100

left(90)  # tourne de 90° dans le sens antihoraire
pensize(1)
forward(50)
penup()  # lève le stylo
right(180)  # tourne de 180° dans le sens horaire
forward(200)
down()  # abaisse le stylo

setheading(0)  # définit l'angle de la tortue (0: Est, 90: Nord, etc.)
forward(50)
right(90)

colormode(255)  # pour pouvoir utiliser des couleurs au format (r, v, b)
ROUGE = (255, 0, 0)
```

```
color(ROUGE)  # définit la couleur du trait ; équivalent à color((255,0,0)
forward(150)
right(90)

fillcolor((0,255,255))  # définit la couleur du remplissage
begin_fill()  # démarre le remplissage
forward(50)
right(60)
forward(100)
end_fill()  # termine le remplissage

hideturtle()  # cache la tortue
```

🐍 **Question 2** : Exécutez le code et analysez chacune des lignes du programme pour en comprendre le rôle.

✍🐍 **Question 3** : Ici, on a utilisé `from turtle import *` pour importer le module `turtle`. Nous avons vu que cette méthode peut potientiellement occasionner des conflits dans le nommage des différentes fonctions.

1. Comment faudrait-il modifier le code si on utilisait à la place `import turtle` ?
2. Et si on utilisait `import turtle as t` ?

⚠ **Remarque importante** : Comme les modules que l'on souhaite écrire dans ce TP vont rester très simples et très courts, on s'autorisera dans la suite à importer le module `turtle` avec :

```
from turtle import *
```

Art-léatoire

On souhaite écrire un module pour générer une oeuvre d'art aléatoire. Pour mieux organiser notre code, on va le décomposer en deux modules :

- `base.py` dans lesquel on va écrire les fonctions pour dessiner les formes souhaitées
- `artleatoire.py` dans lequel on va générer un oeuvre aléatoirement en utilisant (entre autres) les formes du module `base.py`

Création du fichier `base.py`

ℹ Dans ce TP, on se concentre uniquement sur la création d'un carré mais on pourrait imaginer plein d'autres possibilités !

💻 **Question 4** : Créez un fichier appelé `base.py` dans le répertoire du TP puis recopiez-y le code ci-dessous :

```
from turtle import *

def carre(x, y, cote, couleur):
    """
    Dessine un carré.

    Paramètres :
    ------------
        x, y : coordonnées du coin inférieur gauche (int x int)
        cote : longueur du côté du carré (int)
        couleur : couleur du carré (tuple (r, g, b) de trois entiers)
    """
    pass

# Programme principal
colormode(255)
carre(-100, 200, 100, (255,0,0))
```

🐍 **Question 5** : Complétez le code de la fonction `carre` dont la spécification est donnée. Il faudra utiliser convenablement certaines fonctionnalités du module `turtle`.

> Si la fonction est bien écrite, l'exécution du fichier devrait dessiner un carré rouge de côté 100 pixels dont le coin inférieur gauche se trouve aux coordonnées (-100, 200).

Création du fichier `artleatoire.py`

💻 **Question 6** : Créez un fichier `artleatoire.py` dans le répertoire du TP.

🐍 **Question 7** : Dans ce fichier, importez la fonction `carre` du module `base.py` puis exécutez le fichier (qui ne contient que la ligne d'importation pour le moment).

Analyse :

- Bien que seule la fonction `carre` soit importée, tout le contenu de `base.py` est en réalité exécuté et donc son programme principal aussi : comme celui-ci dessine un carré, vous devriez voir se dessiner ce carré en exécutant `artleatoire.py`.
- Ce n'est pas un comportement souhaité, car le programme principal contient généralement des essais et tests qui n'ont pas d'intérêt ailleurs que dans le module où ils sont écrits.
- On peut y remédier : pour que le programme principal d'un fichier ne soit pas exécuté lorsque ce dernier est importé en tant que module, il faut indiquer que le programme principal ne doit être exécuté que si le module est exécuté (et non importé).

🐍 **Question 8** : Remplacez dans `base.py` les lignes

```
# Programme principal
colormode(255)
carre(-100, 200, 100, (255,0,0))
```

par

```
if __name__ == '__main__' :  # il y a des "doubles underscores"
    # Programme principal
    colormode(255)
    carre(-100, 200, 100, (255,0,0))
```

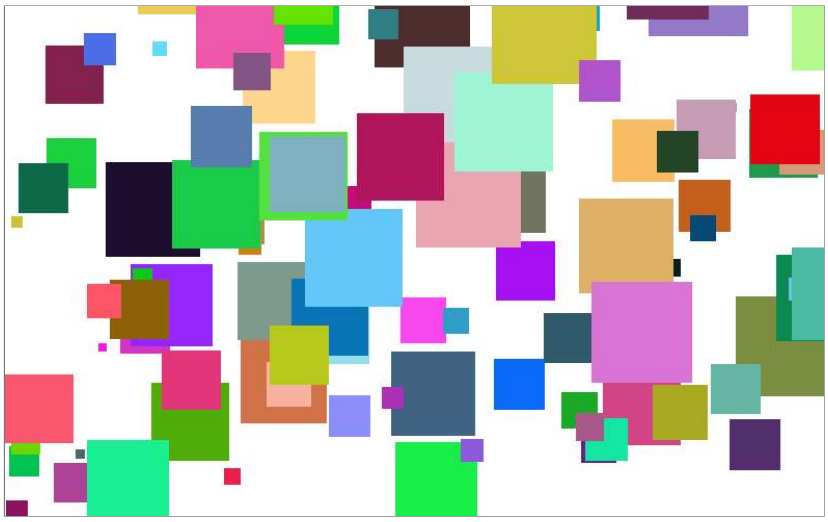
> ℹ️ On ne rentre pas davantage dans le détail ici. Sachez juste que la condition `__name__` == '**main**' est vraie uniquement si le fichier est exécuté et pas s'il est importé. Donc le code du `if` ne sera pas exécuté lorsqu'on importe ce module.

🐍 **Question 9** : Exécutez à nouveau le fichier `artleatoire.py` comme à la question 7, et vérifiez que le programme principal de `base.py` n'est plus exécuté. C'est le comportement souhaité.

🐍 **Question 10** : Dans le module `arleatoire.py`, créez une fonction `peinture(L, H, nb_carres)` qui permet de dessiner avec `turtle` un tableau de dimensions L*H avec un nombre de carrés égal à `nb_carres`. Ces différents carrés doivent être tracés à des positions aléatoires, avec des tailles et couleurs aléatoires.

> **Aide** : Il faudra utiliser à bon escient la fonction `carre` de `base.py` et le module `random`.

Vous devriez obtenir des images de ce genre :



Un tableau de 100 carrés aléatoires.

Parfois le tracé prend un peu de temps, même à vitesse maximale ! Pour voir le rendu final directement, vous pouvez utiliser l'astuce suivante :

```
if __name__ == '__main__':
    fenetre = Screen()
    fenetre.tracer(0)  # permet de désactiver le tracé à l'écran

    # toutes vos instructions de tracé ici

    fenetre.update()  # pour voir le rendu final directement
```

Bilan

Dans ce TP, vous avez :

- créé vos propres modules : `base` et `artleatoire`
- importé et utilisé une fonction du module `base` dans `artleatoire.py`
- importé et utilisé des fonctionnalités des modules `turtle` et `random`

Germain BECKER, Lycée Emmanuel Mounier, ANGERS